อนันต์ กิตติบรรดุ

SOFT MAIL

โปรแกรมสำหรับการเข้ารหัสข้อมูล

การจัดข้อมูลที่ต้องการเก็บเป็นความลับโดยใช้การเข้ารหัสด้วยภาษาซี

ในยุคปัจจุบันข่าวสารและข้อมูลกำลังได้รับความสนใจอย่างสูง และเทคโนโลยีทางด้านนี้เจริญรุดหน้าไปได้รวดเร็วมาก จนเราจะได้ยินกันบ่อย ๆ ว่า โลกเรานั้นแคบลงเรื่อย ๆ ดังนั้นการรักษาความปลอดภัยให้กับข้อมูลจึงมีความสำคัญอย่างยิ่ง โดยเฉพาะข้อมูลที่ต้องเก็บเป็นความลับ เช่น ข้อมูลของบริษัท ข้อมูลของรัฐบาล ข้อมูลด้านการทหาร และอื่น ๆ ซึ่งข้อมูลเหล่านี้จะต้องการเก็บไว้เป็นอย่างดีเพื่อความปลอดภัยและป้องกันการถูกปลอมแปลง หรือถูกเปลี่ยนแปลงข้อมูลจากบุคคลที่ไม่พึงประสงค์ได้โดยเฉพาะข้อมูลในระบบคอมพิวเตอร์ที่อาจมีผู้ใช้อยู่มากมายทั้งภายในองค์กรเอง หรือผู้อื่นที่มาติดต่อกับข้อมูลเหล่านี้ สำหรับการแก้ปัญหาดังกล่าวก็มีด้วยกันหลายวิธี เช่น การนำเรื่องรหัสผ่าน (PASSWORD) มาใช้จำกัดสิทธิความสำคัญของผู้ใช้ การใช้วิธี Copy Protection ดังเช่น เกมส์หลาย ๆ อย่างใช้กันในเครื่อง PC เพื่อกันการก๊อบปี้ แต่วิธีหนึ่งที่ใช้กันมานานและยังเป็นที่นิยมใช้กันอยู่ทุกวันนี้ก็คือ การเข้ารหัสลับข้อมูล (Cryptography หรือ Data Encryption) เพื่อทำให้ข้อมูลถูกเปลี่ยนรูปแบบไปจากเดิมจนไม่สามารถอ่านรู้เรื่อง ยกเว้นผู้ที่รู้รหัสหรือ keyword เท่านั้นจึงสามารถถอดข้อความกลับมาเป็นรูปแบบเดิม

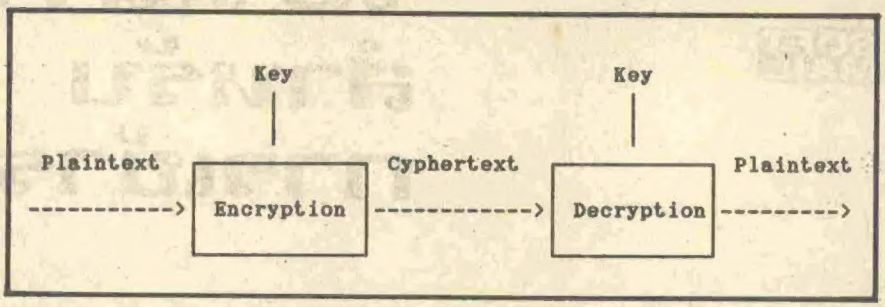
วิธีการทำ Data Encryption นั้นก็มีหลายวิธี ถ้าวิธีใดมีความซับซ้อนสูงก็จะมีความปลอดภัยสูงเช่นกัน สำหรับบนเครื่องใหญ่ ๆ เช่นเครื่อง Super, Mainframe หรือ Mini computer ระบบปฏิบัติการบนเครื่องเหล่านี้เป็นแบบ Multitasking คือสามารถมีผู้ใช้พร้อม ๆ กันได้หลายคน การหาวิธีที่มาใช้สำหรับทำ Data Encryption จึงต้องพยายามทำให้มีความปลอดภัยสูงสุด เช่นการเก็บรหัสผ่าน หรือ PASSWORD เขาก็เก็บไว้ในไฟล์ ๆ หนึ่ง โดยเก็บในรูปแบบที่ผ่าน Data Encryption แล้ว ซึ่งจำต้องใช้วิธีที่มีการคำนวณมาก และทำซ้ำกันหลาย ๆ รอบซึ่งระบบฮาร์ดแวร์ของเครื่องเหล่านี้มีประสิทธิภาพสูง จึงไม่มีปัญหาด้านความเร็วและการคำนวณ แต่สำหรับเครื่องไมโครคอมพิวเตอร์นั้นข้อมูลไม่ได้มีความสำคัญมากมายอย่างเครื่องใหญ่ ๆ แต่ก็ไม่ได้หมายถึงไม่สำคัญเลย ซึ่งวิธีการนั้นก็มีวิธีที่เหมาะสมอื่นอีกที่ใช้ได้ผลดีพอควร และเหมาะสำหรับความสามารถของเครื่องระดับนี้มากโดยที่ไม่ต้องทำการคำนวณมากมายนัก ดังที่คุณจะพบได้ในบทความนี้

หลักการโดยทั่วไป

สำหรับวิธีการเข้ารหัส หรือ Data Encryption นั้น โดยทั่ว ๆ ไปจะเกี่ยวข้องกับข้อมูล 3 ชนิดด้วยกัน ได้แก่ ข้อมูลที่ยังไม่ได้ผ่านการเข้ารหัสเลย ซึ่งเราเรียกกันว่า plaintext, ข้อมูลที่ใช้เป็น key หรือกุญแจ สำหรับ lock ข้อมูล และข้อมูลที่เป็น output ที่ผ่านการเข้ารหัสเรียบร้อยแล้ว ซึ่งเรียกว่า ciphertext

และส่วนของการถอดรหัส หรือ Data Decryption ก็จะเหมือนกัน แต่ข้อมูลที่เป็น plaintext ของวิธี Data Encryption จะเป็น ciphertext แทน และ output ที่ได้ออกมาจะเป็น plaintext ตามเดิม โดยต้องใช้ค่า key เหมือนกันด้วยสรุปได้ดังรูปที่ 1

สำหรับ Algorithm ที่นิยมใช้กันในเครื่องระดับพีซีก็มีด้วยกัน 2 แบบ คือ แบบ Exclusive-or ระหว่าง plaintext กับคีย์ที่ใช้เข้ารหัส วิธีเป็นวิธีที่ง่าย ซึ่งวิธีนี้จะให้ความปลอดภัยสูงสุด หากคีย์ที่ใช้มีความยาวมากกว่า plaintext ขอให้ดูตัวอย่างดังรูปที่ 2 และแบบ Data Encryption Standard หรือที่เรียกกันสั้น ๆ ว่า DES Algorithm อันเป็นวิธีการมาตรฐานที่ใช้กันอย่างมากที่สุด และได้รับการยอมรับให้เป็นมาตรฐานจาก National Bureau of Standard (NBS) ว่าก็มีความปลอดภัยสูงมาก วิธีนี้จะใช้การคำนวณมากและต้องทำซ้ำ ๆ กันเป็นจำนวนถึง 16 รอบ โดยมองข้อมูลเป็นบล็อก ๆ ละ 64 บิต และใช้คีย์ซึ่งมี 64 บิตเช่นกันมาผ่านฟังก์ชันที่มีการคำนวณซับซ้อนมาก ciphertext ที่ได้จึงมีความปลอดภัยสูง จึงยากที่ผู้อื่นจะมาถอดรหัสได้หากไม่รู้ค่าคีย์ว่าเป็นอะไร



รูปที่ 1 แสดงการทำงานการเข้ารหัสและการถอดรหัส

```
char * crypt_txt;  /* string of keyword for data encryption */
int crypt_idx;     /* index to string   */
void Crypt(unsigned char *buff)
/*
     example for data encryption by exclusive-or(xor)
     encrypt data from string buff if buff is "PLAINTEXT" or ..
     decrypt data from string buff if buff is "CYPHERTEXT"
     this function must be called till reach end of buff string
 */
{
     *buff ^= crypt_txt[crypt_idx]
     /*
     update crypt_idx to next char of keyword for next calling
     since crypt_txt key length is often less than input data so ..
     reuse it by wrap around again
     */
     if (++crypt_idx == crypt_len) crypt_idx = 0;
}
```

รูปที่ 2 แสดงตัวอย่างการ Data Eneryption โดยวิธี Exclusive-or

และสำหรับวิธีที่ผมใช้ในบทความนี้เป็นวิธีแรก แต่ได้แก้ข้อเสียของวิธีแรกคือเรื่องของคีย์ จะเห็นว่าข้อมูลที่เราจะใช้เข้ารหัส หรือ plaintext นั้นหากมีขนาดมาก ๆ คงไม่ไหวแน่ ๆ ที่จะต้องใช้คีย์ที่มีมากตามไปด้วย แต่วิธีผมจะใช้คีย์ขนาดสัก 5, 6 ตัวก็พอ โดยจะนำคีย์มาใช้ใหม่อีก คือใช้แบบเวียนเทียนเลย หลังจากคีย์เหล่านี้ถูกใช้ครบทุกตัวแล้ว และได้เพิ่มความสามารถอื่น ๆ อีก (จากรูทีน Crypt( ) ในโปรแกรม "Encrypt.c") สำหรับการใช้งานนั้นผมทำไว้ 2 เวอร์ชัน คือ โปรแกรม "Excrypt.c" และ โปรแกรม "Cryptpro.c" ดังรูปที่ 3 และ 4 เวอร์ชันแรกนั้นให้ดูการใช้งานแบบง่าย ๆ โดยจะอ่าน input file (Plaintext) และ key เข้ามา แล้วเรียกรูทีน Crypt( ) เพื่อทำการ encrypt ทีละไบต์ โดยครั้งแรกจะได้ plaintext ออกมา และถ้านำ plaintext นี้ไปเป็น input file และใช้คีย์เดิมเรียกซ้ำอีกครั้งจะได้ข้อมูลกลับมาเหมือนเดิม สำหรับเวอร์ชันหลังนั้นผมทำให้ซับซ้อนมากขึ้น ซึ่งเชื่อว่าปลอดภัยสูงพอสมควรลักษณะเด่นของโปรแกรมนี้จะตรวจคีย์ก่อนด้วยว่ามีความแตกต่างกันอย่างน้อยมากกว่า 5 (เป็นค่า MINKEY ที่ define ไว้ในโปรแกรม cryptpro.c) ตัวขึ้นไปด้วยจึงจะทำงานให้ และรับ option เข้ามาว่า จะทำการ

Encrypt หรือ Decrypt สำหรับรายละเอียดของการทำงานขอให้ศึกษาจากซอร์สโค้ดที่ให้ไว้ได้ ทั้งนี้ผมได้แทรก comment ไว้มากพอสมควร เชื่อว่าจะอ่านทำความเข้าใจได้ไม่ยากนัก

อย่างไรก็ดี อาจกล่าวได้ว่าไม่มีระบบเข้ารหัสวิธีใดจะไม่สามารถทำลายได้ แต่จุดประสงค์หลักของการทำ Data Encryption ก็เพื่อมิให้มาถอดรหัสกันได้อย่างง่าย ๆ หากไม่รู้คีย์และทำให้ผู้ที่พยายามจะถอดรหัสต้องเสียทั้งเวลาและค่าใช้จ่ายสูงมาก จนอาจต้องใช้วิธีซื้อตัวพนักงานของบริษัทคู่อริมาใช้ยังคุ้มกว่าเสียอีก แต่วิธีในบทความนี้คงไม่ถึงขั้นนั้นหรอกครับ เพียงทำให้ผู้อ่านได้รู้จักกับเรื่องการทำ Data Encryption ความสำคัญ และวิธีการโดยคร่าว ๆ และอาจนำรูทีนเหล่านี้ไปเป็น tools สำหรับการพัฒนาโปรแกรมขึ้นมาใช้เอง เช่นใช้ encrypt password หรือ data ที่จะเก็บในโปรแกรม หรือใช้ในการส่งข้อมูลผ่านทางโมเด็ม เพื่อกันการถูกดักอ่าน ถ้าข้อมูลนี้ไม่อยากให้ใครอื่นรู้

รูปที่ 3 โปรแกรม Encrypt.c

```
/***
        Encrypt.c  - Encrypt/Decrypt data

        Programmer - Anan Kittibunlu
        Created at - 16/Dec/1990

 **/
£include <stdio.h>
£include <string.h>

/* Global Data */
char *crypt_txt;         /* input text string key to encrypt */
int   crypt_idx = 0;     /* index to point cryp_txt          */
int   crypt_len = 0;     /* length of string key             */

void SetCryptTxt(char *txt) {
     crypt_txt = (char *) txt;
     crypt_len = strlen(txt);
}

void Crypt(unsigned char *buff)
/*
     encrypt data from string buff if buff is "PLAINTEXT" or ..
     decrypt data from string buff if buff is "CYPHERTEXT"
     this function must be called till reach end of buff string
 */
{
     /* change data string one item to encrypt/decrypt by exclusive-or method */
     *buff ^= crypt_txt[crypt_idx] ^ (crypt_txt[0]*crypt_idx);
     /*
     since crypt_txt key length is often less than input data so ..
     reuse it by wrap around again (circular) and for security
     change crypt_txt key every time after it be used
     */
     crypt_txt[crypt_idx] += ((crypt_idx < (crypt_len-1)) ?
                              crypt_txt[crypt_idx+1] : crypt_txt[0]);
     if (!crypt_txt[crypt_idx]) crypt_txt[crypt_idx]++; /* sure not zero */
     if (++crypt_idx == crypt_len) crypt_idx = 0;
}
```

รูปที่ 4 โปรแกรม Excrypt.c

```
/**
     excrypt.c   - example test routine encrypt/decrypt

     Programmer  - Anan Kittibunlu                                    →
```

รูปที่ 4

```
    Created at  - 17/Dec/1990
**/

#include <stdio.h>
#include <stdlib.h>

extern void Crypt(unsigned char *);  /* from encrypt.c */
extern void SetCryptTxt(char *);     /* from encrypt.c */

#define  BUFFSIZE  1024

void main(int argc, char *argv[])
{
    int nread;
    unsigned char buff[BUFFSIZE]
    FILE *fin, *fout;

    if (argc != 4) {
       printf(" Usage : %s input output key(encrypt/decrypt) \n",argv[0]);
       exit(1);
    }
    if ((fin = fopen(argv[1],"rb")) == NULL) {
       printf("Can't open input-file: %s \n",argv[1]);
       exit(1);
    }
    if ((fout = fopen(argv[2],"wb")) == NULL) {
       printf("Can't open output-file: %s \n",argv[1]);
       exit(1);
    }
    SetCryptTxt(argv[3]);
    while ((nread = fread(buff, sizeof(char), BUFFSIZE, fin)) > 0) {
       /*
        read file from file fin each BUFFSIZE, if end-of-file
        nread = 0
        */
       unsigned char *sptr;
       int i;

       /* assign sptr to point to buff(sptr is a freedom pointer) */
       /* since buff is const pointer so we can't change it */
       sptr = buff;
       for (i=0; i<nread; i++) Crypt(sptr++); /* Crypt until nread(bytes)  */
       fwrite(buff, sizeof(char), nread, fout);
    }
    fclose(fin); fclose(fout);
    exit(0);
}
```

โปรแกรม Cryptpro.c

```
/**
    cryptpro.c  - routine encrypt/decrypt for professional security

    Programmer  - Anan Kittibunlu
    Created at  - 17/Dec/1990
**/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <io.h>
```

```
extern void Crypt(unsigned char *);  /* from encrypt.c */
extern void SetCryptTxt(char *);     /* from encrypt.c */

#define  BUFFINSIZE  1024
#define  BUFFOUTSIZE 32
#define  KEYMIN    5
#define  OK        0
#define  ERROR    -1

union {
    unsigned char buffin[BUFFINSIZE];
    unsigned char buffout[BUFFOUTSIZE][BUFFOUTSIZE];
    /* buffin and buffout use same memory, but ..*/
    /* buffin will see only one dimension        */
    /* buffout will see two dimension            */
}buff;

int check_cryptkey(char *cryptkey) {
/*
    to check encrypt/decrypt key that have strlength more than KEYMIN
    and those key must have distince more than KEYMIN or not
    if yes return OK, else ERROR to return
 */
    int i,j;
    static char found[KEYMIN+1];  /* initialized to zero */
    int cryptlen = strlen(cryptkey);

    if (cryptlen < KEYMIN) return ERROR;
    for (i=0,j=0; i<cryptlen; i++) {
        /*
           to search string "found" from cryptkey char
           if "found" string iterate it will be skip
           to next cryptkey char
         */
       if(strchr(found,cryptkey[i])) continue;
       found[j++] = cryptkey[i];
       if (j == KEYMIN) return OK;
    }
    return ERROR;
}

void encrypt(FILE *fin,FILE *fout) {
/*
     these routine will change plain text to cypher text by..
     fouth first bytes of fout to keep size of plain text another keep
     real cypher text
     and for really professional security we will not transform input to
     crypt to output by sequence stream(BUFFINSIZE) but to devide it is
     row and col which equally and to fill by column stream (BUFFOUTSIZE)
 */
    int nread;
    int fd;
    long fsize;

    fd = fileno(fin);  /* to change from file stream to file handle no */
    fsize = filelength(fd);
    fwrite(&fsize, sizeof(long), 1, fout); /* keep length of plaintext */
    srand((unsigned) fsize);
    while ((nread = fread(buff.buffin, sizeof(char), BUFFINSIZE, fin)) > 0) {
       int i,j;

       /* case have data left, created those data by random  */
       /* since this method must use full BUFFSIZE plaintext */
```

```
        while (nread < BUFFINSIZE) buff.buffin[nread++] = rand();
        for (i=0; i<BUFFOUTSIZE; i++)
          for (j=0; j<BUFFOUTSIZE; j++)  Crypt(&buff.buffout[j][i]);
        fwrite(buff.buffin, sizeof(char), BUFFINSIZE, fout);
    }
}

void decrypt(FILE *fin,FILE *fout) {
    long  fsize;
    int nread;

    fread(&fsize, sizeof(long), 1, fin);
    while ((nread = fread(buff.buffin, sizeof(char), BUFFINSIZE, fin)) > 0) {
       int i,j;
       for (i=0; i<BUFFOUTSIZE; i++)
         for (j=0; j<BUFFOUTSIZE; j++)  Crypt(&buff.buffout[j][i]);
       if (fsize >= BUFFINSIZE)
         fwrite(buff.buffin, sizeof(char), BUFFINSIZE, fout);
       else
         /* write file only fsize(sizeof plaintext)       */
         /* since other is garbage that come from random */
         fwrite(buff.buffin, sizeof(char), (int)fsize, fout);
       fsize -= nread;
    }
}

void main(int argc,char *argv[])
{
    FILE *fin,*fout;

    if (argc != 5) {
       printf("Usage: %s Option input output key(encrypt/decrypt) \n",argv[0]);
       puts("Option  -E or /E is encrypt");
       puts("        -D or /D is decrypt");
       exit(1);
    }
    if ( (argv[1][0] != '-' && argv[1][0] != '/') ||
         (toupper(argv[1][1]) != 'E' && toupper(argv[1][1]) != 'D') ) {
       puts("Unknow Option");
       exit(1);
    }
    if ((fin = fopen(argv[2],"rb")) == NULL) {
       printf("Can't open input-file: %s \n",argv[2]);
       exit(1);
    }
    if ((fout = fopen(argv[3],"wb")) == NULL) {
       printf("Can't open output-file: %s \n",argv[3]);
       exit(1);
    }
    if (check_cryptkey(argv[4]) == ERROR) {
       puts("Your key is not security..! ");
       puts("Try again");
       exit(1);
    }
    SetCryptTxt(argv[4]);
    if (toupper(argv[1][1]) == 'E')
       encrypt(fin,fout);
    else
       decrypt(fin,fout);
    fclose(fin); fclose(fout);
    exit(0);
}
```